AFRL-AFOSR-UK-TR-2010-0014

Sensor Fusion, Prognostics, Diagnostics and Failure Mode Control for Complex Aerospace Systems

Paul Stewart

University of Salford
School of Computing, Science and Engineering
Newton Building
Greater Manchester, United Kingdom M5 4WT

EOARD GRANT 09-3058

October 2010

Final Report for 27 March 2009 to 27 September 2010

Air Force Research Laboratory
Air Force Office of Scientific Research
European Office of Aerospace Research and Development
Unit 4515 Box 14, APO AE 09421

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
25-10-2010	Final Report	27 March 2009 – 27 September 2010

4. TITLE AND SUBTITLE

Sensor Fusion, Prognostics, Diagnostics and Failure Mode Control for Complex Aerospace Systems

5a. CONTRACT NUMBER

FA8655-09-1-3058

5b. GRANT NUMBER

Grant 09-3058

5c. PROGRAM ELEMENT NUMBER

61102F

6. AUTHOR(S)

Professor Paul Stewart

5d. PROJECT NUMBER

5d. TASK NUMBER

5e. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of Salford
Newton Building
Greater Manchester M5 4WT
United Kingdom

8. PERFORMING ORGANIZATION REPORT NUMBER

Grant 09-3058

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

EOARD
Unit 4515 BOX 14
APO AE 09421

10. SPONSOR/MONITOR'S ACRONYM(S)

AFRL/AFOSR/EOARD

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

AFRL-AFOSR-UK-2010-0014

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This report results from a contract tasking University of Salford as follows: This project examined the speed-up of search for hardware-in-the-loop based searches. The details for carrying out parallel and distributed computations with the search system are conveyed and used to perform large searches. Two methods of parallelization are available due to the two loop feature of the search. In the first method, the master processes each iteration of the outer loop and subsequent inner loops serially. The slaves are used to evaluate the fitness values of the inner loop individuals in parallel. The result of this arrangement is that the processing is a form that is analogous to standard parallel genetic algorithms, and as such existing parallel methodologies can be used. The second method involves the master processing the outer loop only; the inner loop for each outer individual is processed in its entirety by a slave. Hence, outer loop individuals are processed in parallel. The advantage of this second method is that communication overheads are reduced by a factor of G_i * X_i. This is because communication between master and slaves only occurs at each iteration of the outer loop, rather than at each iteration of the inner loop, as in the first method. A methodology for minimizing the number of trials when using hardware-in- the-loop to direct the search is presented. An experiment is conducted demonstrating that control solutions developed using models under simulation do not always perform as expected in hardware. The demonstration shows a controller developed from a simplified model for an Electric Thrust Reversal Actuation System (ETRAS), that when implemented in hardware does match the performance under simulation. This is because the controller is fundamentally based upon the accuracy of the model from which it has been derived. It is sometimes the case that very accurate models are not available and therefore it is desirable to use hardware-in-the-loop to direct the search. The main problem with hardware-in-the-loop is that performing hundreds or thousands of trials is often unfeasible due to time constraints. To minimize the number of trials needed, pre-selection is introduced that uses a hybrid of simulation and hardware-in-the-loop trials to direct the search. The fitness of each individual is calculated using simulations in the first instance; if an individual performs better than a set threshold, then its fitness value is re-evaluated with hardware-in-the- loop. The search algorithm then uses the hybrid of fitness data to rank the results. The ETRAS controller is developed using pre-selection, showing that a controller can be found that matches the performance requirements. The threshold values were established through trial and error, through varying the values and observing whether the search was converging. Ultimately, out of 180,000 fitness calculations carried out under simulation, only 11,301 (6.3%) were re-evaluated with hardware-in-the-loop. This is a significant reduction in experimental trials making hardware-in-the-loop searches feasible.

15. SUBJECT TERMS

EOARD, Distributed Systems, Computational Modeling, Neural Networks

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18, NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			JAMES LAWTON Ph. D.
UNCLAS	UNCLAS	UNCLAS	SAR	36	**19b. TELEPHONE NUMBER** *(Include area code)* +44 (0)1895 616187

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39-18

Sensor Fusion, Prognostics, Diagnostics and Failure Mode Control for Complex Systems

Professor Paul Stewart
Head of School of Engineering
University of Lincoln UK

Introduction

There is currently a significant gap in the knowledge of basic science relating to the prognostics, diagnostics and catastrophic fault-tolerance of complex engineering systems which are often composed of multiple inter-connected sub-systems. In essence, future air vehicles will be extremely complicated, but current theory isn't up to the task of optimally controlling them, particularly in terms of predicting failure, and making the system robust to sub-system failure. In order to design well for these complex systems, it is necessary to take on board all the potential interactions between the vehicle subsystems, and in particular, predicting what happens when one goes down. This is computationally very intensive, and results in potential combinatorial explosion (in terms of the combinations of variables involved in an optimal design) during analysis and algorithmic design. In this research project we have combined our previous work on complex system simulation, analysis and design, with the work on Modeling Field Theory (MFT) developed by Leonid Perlovsky AFRL/RY, published in Neural Networks and Intellect (Oxford University Press 2001, ISBN 0-19-511162-1). This is a promising methodology to bridge the gap in the basic complex systems theory. A multiple-model adaptive approach as described by Perlovsky's MFT would be the most appropriate to drive the learning of fault-tolerant stable systems, and also the predictive models necessary to perform prognostics and diagnostics. The use of a fuzzy approach, specifically the use of Sensor Fusion MFT, will avoid the combinatorial explosion effects which are to be expected in the design of complex systems.

The approach taken has been to investigate the basic concepts and develop a theory to describe sensor system architectures that will allow robust/optimal analysis for distributed complex electrical generation and actuation systems. This is particularly important research which addresses the exponential rise in the number of sensors available on future aircraft systems which are embedded in flight and electrical power systems. This exponential growth offers unprecedented opportunities to predict, analyse and mitigate failure in aircraft systems and sub-systems. The theory has also been extended to include safety-criticality, diagnostics and prognostics and failure mode survivability. In particular, we have examined the following properties: Fault tolerance; Dynamic stability; Predictable interactions in complex systems; Catastrophic failure: reconfigurable systems; and Complex system prognostics and diagnostics.

The motivation for this project is relatively straightforward. Two core issues complicate making decisions based upon the information from multiple sensors. Firstly, real-life sensor data is often noisy; secondly the methodologies for searching the data are often swamped by either the combinatorial explosion of a multi-dimensional space, or the sheer volume of incoming data. The project was thus split into two 6-month parts, each of which produced an interim report. This final report combines each of those reports into a single document.

Sensor Fusion, Prognostics, Diagnostics and Failure Mode Control for Complex Systems – 6 Month Report

Professor Paul Stewart
Head of School of Engineering
University of Lincoln UK

Introduction

The motivation for this project is relatively straightforward. Two core issues complicate making decisions based upon the information from multiple sensors. Firstly, real-life sensor data is often noisy; secondly the methodologies for searching the data are often swamped by either the combinatorial explosion of a multi-dimensional space, or the sheer volume of incoming data.

The project is thus split into two parts, and this report describes the work conducted during the first six-month phase to address the problem posed by noisy sensor data. The author has previously conducted research into the development of techniques to improve the performance of several standard search heuristics such as gradient descent, variable neighbourhood search and simulated annealing in searching increasingly noisy data surfaces [Stewart 2010], a copy of which is attached to this report. One of the most significant aspects of data search with heuristics is the concept of 'no free lunch', that is, a heuristic which performs well with one particular data stream is not guaranteed to perform well with another data stream, and often require significant *a priori* knowledge and manual tuning. With this in mind, the project overall will focus on Genetic Algorithms and Genetic Programming, with the aim of developing methodologies for the automatic creation of models for the interpretation of multiple sensor data from real-time control systems, both under simulation and with hardware-in-the-loop. In particular, the ability to raise the level of generality by removing the choice of model structure as an initial parameter is considered.

It is common for system analysts to choose model structures based on their knowledge and experience, and to then tune that model using either classical or heuristic methods. Removing this input provides the opportunity to create a hybrid metaheuristic search that produces both the model topology and tunes the associated model parameters. Furthermore, it provides a means to apply a hyperheuristic methodology for the automatic creation of models.

Genetic programming (GP) has already been shown to produce controllers and models that are human-competitive, moreover, producing solutions that infringe on previous patents showing that GP can be used as an inventing machine [Koza 2000]. However, the approach taken evolves topologies and component values together, creating a very large single search space. This approach is computationally intensive and in terms of accelerating convergence, does not offer many immediate options other than parallelisation. Accepting the fact that finding a solution is primarily based upon first discovering the correct topology, it appears that there is opportunity to split the process into two. The concept investigated in this work is to produce candidate model topologies using a novel algorithm and to then tune the candidates individually using known metaheuristics. As will be shown in a later report, this approach provides greater opportunity to apply novel acceleration methods in addition to parallelisation.

The work described in this initial report, represents the necessary foundation work performed to develop search methodologies which perform well even in the presence of the significant measurement noise which is present in all sensor measurements, and which, if unaddressed can significantly affect the performance of heuristic methods searching multiple sensor data models. Subsequently, the performance of Genetic algorithms in the context of noisy data sources will be addressed.

Context

The author has previously investigated a methodology which makes use of the data evaluated by the heuristic during the search, and utilises this to generate response surfaces. These response surfaces are used to generate probability surfaces to provide the search heuristic with weighted stochastic decision support (figure 1).

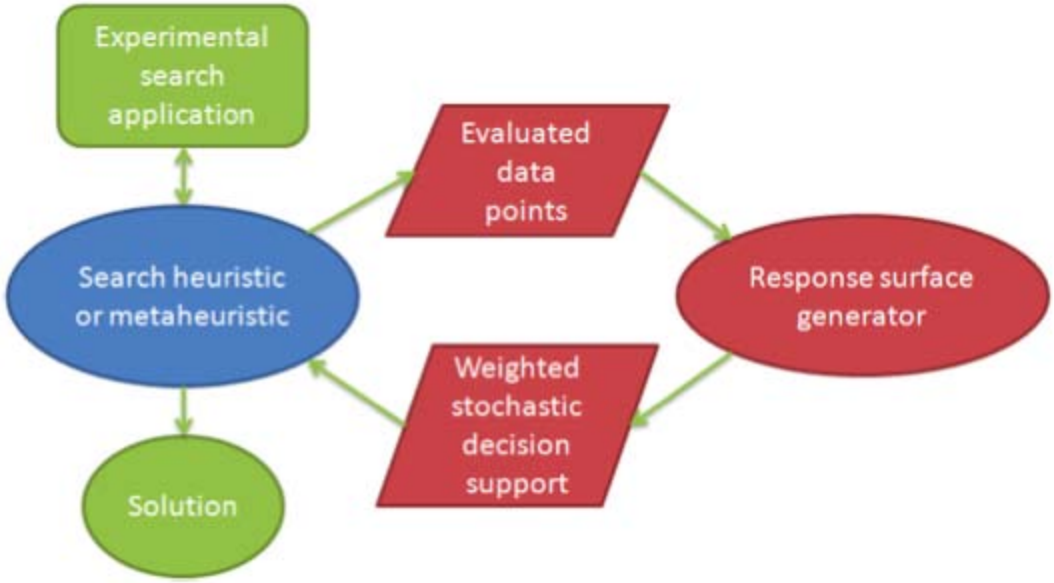

Figure 1 *Decision support methodology*

A weighted stochastic decision support (WSDS) operator based on response surfaces (RS) was designed, which supports the heuristic and guides the experimental process to predicted areas of interest in the search space. The methodology was developed on a 3-dimensional data surface with multiple local-minima and large basin of attraction, and varying levels of noise. Basic gradient descent (GD), simulated annealing (SA) and variable neighbourhood search (VNS) were considered, since in most of their varieties, implementation is simple, and basic tuning rules are readily available, making them commonly used heuristics in the experimental community. Basic gradient descent, SA and VNS were supplemented by the WSDS methodology, and performance compared to the basic form of the metaheuristics. The supplemented metaheuristics were shown to have significantly improved performance when searching over increasingly noisy surfaces.

The application of the WSDS method in its basic form is illustrated in figure 2. The heuristic uses the initial random starting point to initiate its search, storing each point that is evaluated into an array containing the most recent search data.

The response surface can take any arbitrary order. If the metaheuristic finds a minimum that is below a preset value, then the procedure terminates. Otherwise, a weighted random jump is made to escape the local minimum based upon the support surface.

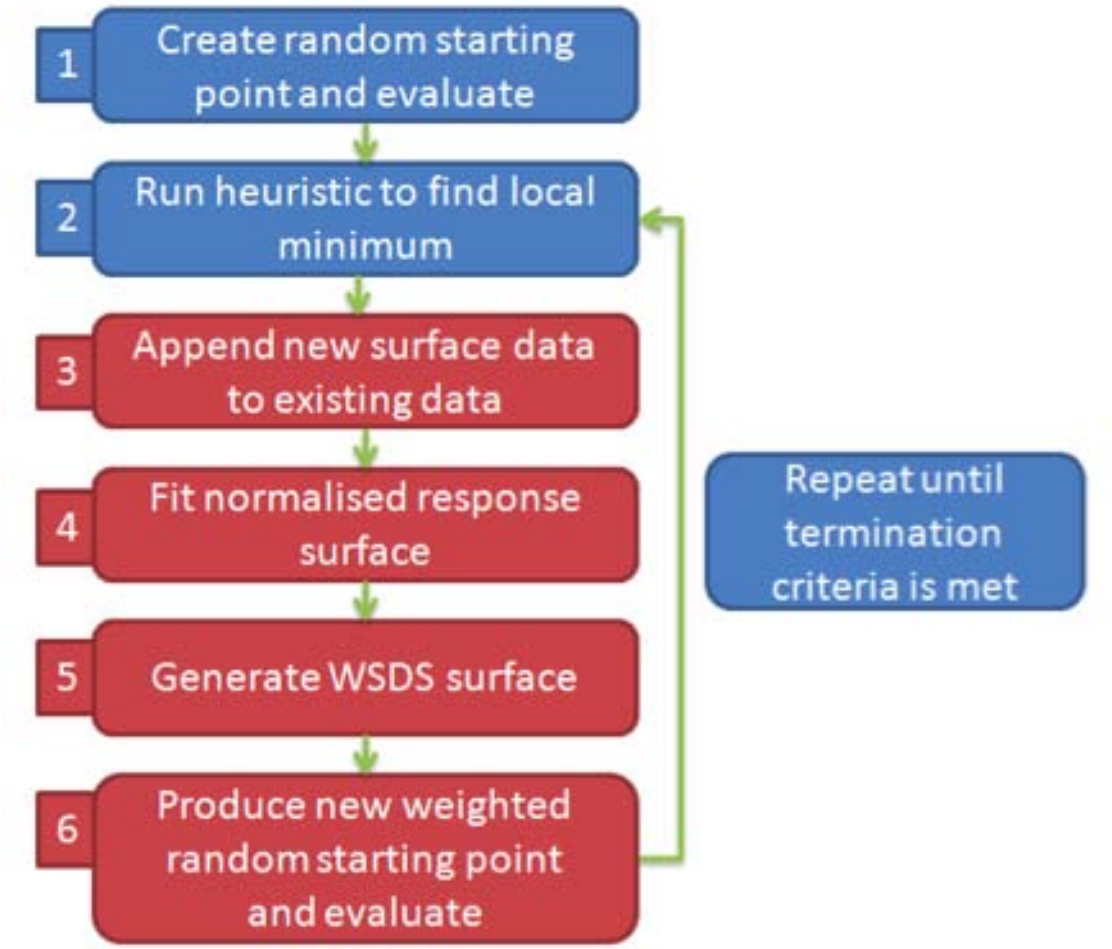

Figure 2 *Diagram showing basic WSDS method combined with heuristic*

For the development of this methodology, a realistic data surface with multiple local minima, plateaux and one global minimum is considered. The standard MATLAB *peaks* surface (figure 3) describes a combinatorial process in two variables.
In order to investigate the effects of noise, progressively larger amounts of Gaussian noise are added to the smooth surface (peaks0) to give peaks 1,2,3 (figure 3 – figure 6). The magnitude of the noise is given as a fraction of the range of values of this input array. The addition of the noise is achieved by utilising the R function *jitter* written by Werner Stahel and Martin Maechler, ETH Zurich. The *jitter* function adds a small amount of Gaussian (white) or uniform noise to a vector, matrix or *N-D* array. This function is ideal for adding noise to a signal for processing, or generating starting conditions for chaotic functions. The magnitude of the noise is given as either a fraction of the smallest difference between values of the input array, or as a fraction of the range of values of this input array.
In order to examine the effectiveness of the method, another search space is introduced, namely the *bump* problem [Keane 1994], which is a smooth surface

comprising many peaks, all of a similar size. Also the optimal value is defined adjacent to a constraint boundary. It has been noted that these features render it relatively difficult for most optimisers to deal with [Keane 1995] (Figures 7&8).

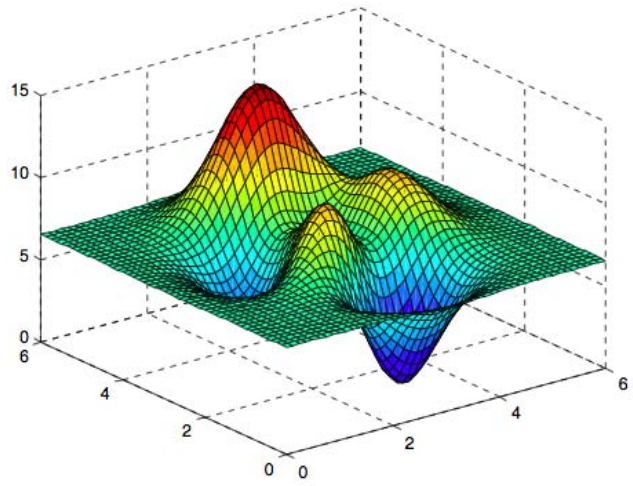

Figure 3 *Smooth algorithm development fitness landscape: peaks0*

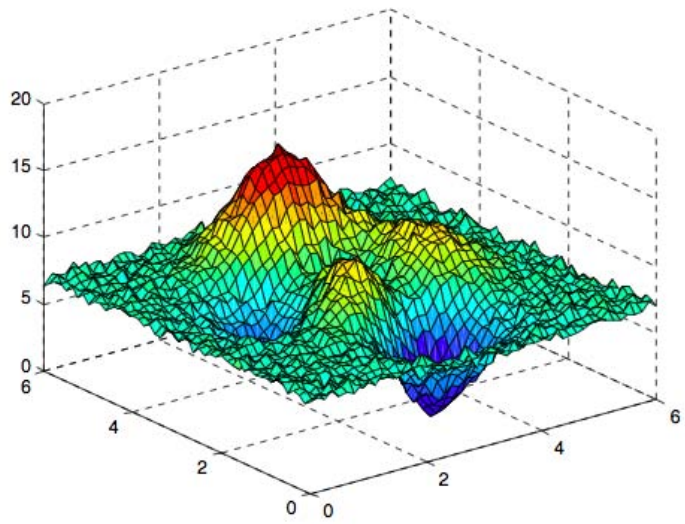

Figure 4 *Rugged algorithm development fitness landscape: peaks1*

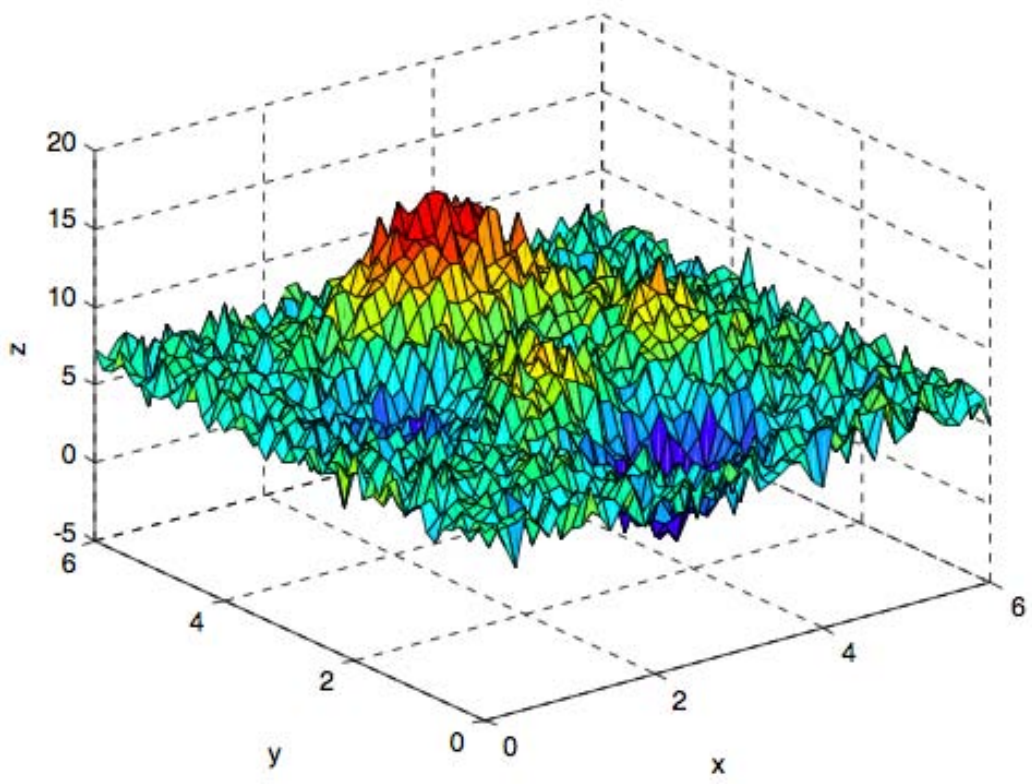

Figure 5 *Rugged algorithm development fitness landscape: peaks2*

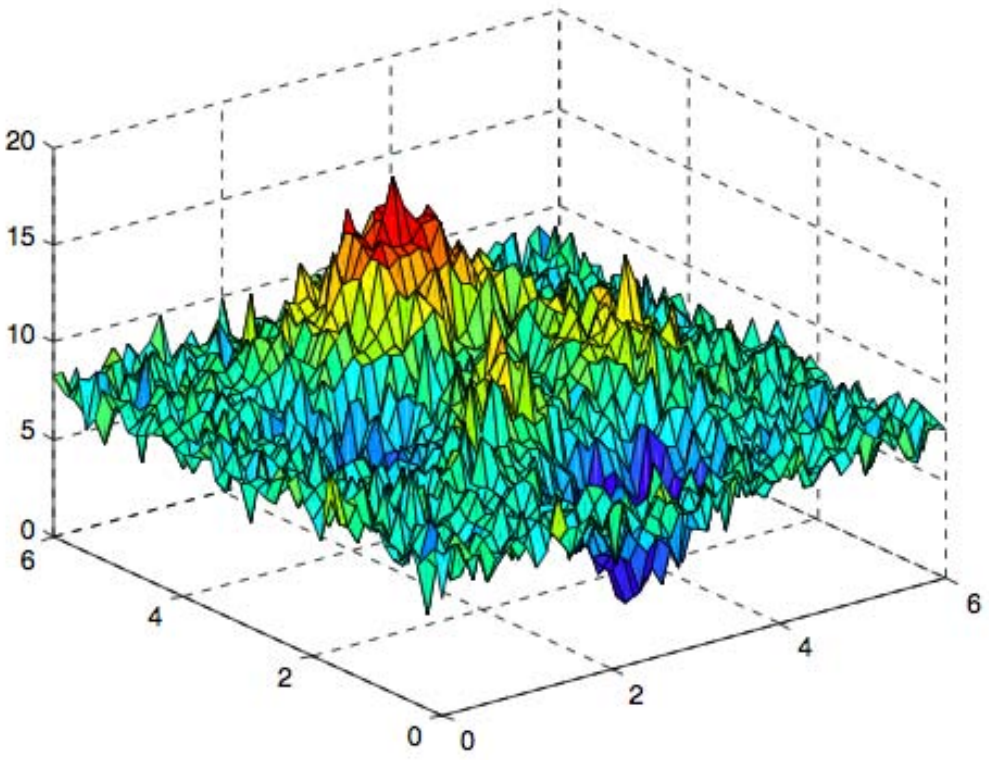

Figure 6 *Rugged algorithm development fitness landscape: peaks3*

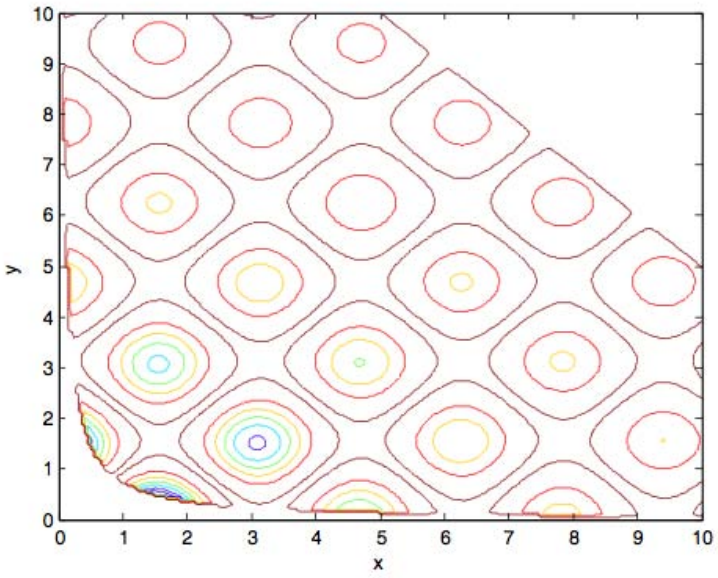

Figure 7 *Contour map for two-variable bumps function*

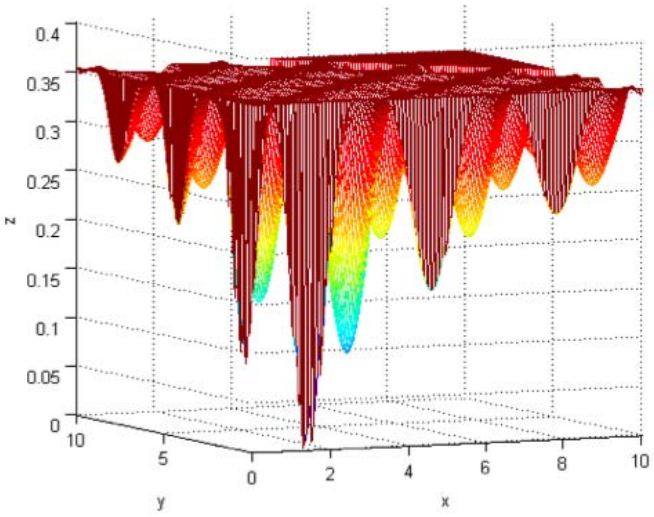

Figure 8 *Two-variable bumps function surface*

We will now consider the novel combination of the WSDS methodology with Genetic Algorithm search

1. Genetic Algorithms

In Genetic Algorithms, suitable solutions to a problem are found through the manipulation of a population of candidate solutions; a candidate solution is termed an *individual* or *chromosome*. Problem variables are encoded into either a string of binary digits, integers, or floating-point numbers. Each individual in a population is evaluated to give its fitness, with which they are then ranked in order of best solution. The higher-ranking individuals are then selected and recombined to form a new population (next generation). Probabilistic mutation

occurs according to a *mutation rate*. in the case of a binary representation this typically involves flipping binary digits. This process repeats, and through successive generations, good traits start to dominate the population, which results in an increase of quality of solutions. A successful GA depends on the correct choice of population size, number of generations and genetic operator parameters [Mitchell 1996].
GAs have been shown to be especially suited to multi objective optimisation [Fonseca 1993], in particular, multi objective genetic algorithms (MOGAs) are being widely used to tune controllers [Fonseca 1994]. Such is the widespread use, there are now a number of software packages available to run MOGAs, one such offering is The MATLAB Genetic Algorithm Toolbox created by Chipperfield and Fleming [Chipperfield 1995].

1.1 Population Representation and Initialisation

An individual in a population can either be represented as a binary string, integer or a floating point value. The most common method is to encode variables as binary strings, which are then concatenated to form a chromosome. However, the choice of representation is context specific, for example, it is argued that binary representation obscures the nature of the search [Bramlette 1991], and that real-valued encoding offers a number of advantages in numerical function optimisation [Wright 1991, Jaskiewicz 1998]. However, the preference for using binary representations of solutions in genetic algorithms is derived from schema theory [Holland 1975], which analyses genetic algorithms in terms of their expected schema sampling behaviour, under the assumption that mutation and recombination are detrimental. The use of Gray encoding [Caruana 1988], or logarithmic scaling [Schmitendorf 1992] are techniques recommended to improve binary encoding representations. For the purpose of this work, operators will be described in terms of binary representation as this is the representation that is used throughout the project.
The initialisation of the population is commonly achieved by generating individuals in the population using a random number generator. Other methods include the extended random initialisation procedure [Bramlette 1991], and seeding the population with known good individuals [Burke 2000]. It is generally accepted that a random initialisation performs equally as well, or better than more sophisticated methods, this is attributed to the increased diversity of the initial population. Furthermore, alternate methods often require *a priori* knowledge of the problem, or need to be used in conjunction with knowledge based systems.

1.2 Objective and Fitness Functions

To compare the performance of individuals in a population, a measure needs to be taken of how well they performed in the problem domain. The *objective function* is used for this purpose, providing a value for each individual that can be easily compared. In the case of a minimisation problem, the individual with the lowest value will signify that it performs best, also known as the most *fit*. To determine the relative performance of individuals in a population, a fitness function is used that transforms the objective value into a measure of relative fitness. A commonly used transformation is that of proportional fitness

assignment, whereby the individual fitness of each individual is calculated as the individuals raw performance relative to the whole population. This method leads to each individual possessing a probability of reproducing according to its relative fitness. An alternate transformation assigns a fitness value according to the rank of the individuals in the population [Baker 1987]. This method avoids highly fit individuals dominating in the early generations, as is the case with the previous linear scaling approach.

1.3 Selection Methods

Selection is the process in which individual genotypes are chosen from a population for later breeding. Many methods of selection are based primarily on a *roulette wheel* mechanism (figure 9), to probabilistically select individuals based on a measure of their performance. In figure 9, the size of each segment corresponds to the fitness value of the associated individual, with the circumference of the wheel being the sum of all fitness values. To select an individual, a random number is generated in the interval from zero to the circumference of the wheel. The individual whose segment spans that number is selected. This process is repeated, until the desired number of individuals has been selected. This selection method is defined as stochastic sampling with replacement (SSR). The segment size remains the same throughout the process and hence the probability of selecting an individual remains the same. There are variations to this method that alter the probability of selecting the individuals during the process, which include Stochastic Sampling with Partial Replacement (SSPR), and Remainder Stochastic Sampling with Replacement (RSSR) [Goldberg 1991].

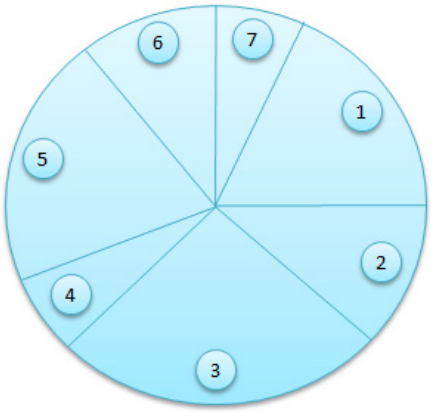

Figure 9 *Roulette wheel mechanism showing different intervals for each individual*

There are, however, two other methods worthy of note that do not use the roulette wheel mechanism, these are Stochastic Universal Sampling (SUS) and tournament selection. SUS provides zero bias and minimum spread [Baker 1987]. Where *bias* is the difference between an individual's normalised fitness and its expected probability of reproduction, and *spread* is the range of possible values for the number of offspring of an individual. The individuals are mapped to contiguous segments of a line, such that each individual's segment is equal in size to its fitness, exactly as in roulette wheel selection. Equally spaced pointers

are then placed over the line, as many as there are individuals to be selected. The position of the first pointer is chosen by a random number from zero to the distance between pointers. The positions of the pointers then define which individuals are selected.
In basic tournament selection [Goldberg 1991], a number of individuals (tournament size) are chosen randomly from the population to compete, the best individual from this group is selected as a parent. This process is repeated for the number of parents that are required. If the tournament size is large, then weak individuals are less likely to be selected.

1.4 Single-point and Multi-point Crossover

The basic operator for recombination, or producing new chromosomes, is that of crossover. The simplest form of crossover is that of single-point crossover (figure 10). After selection has taken place, two parents are used to create two children by randomly choosing a position to split both the chromosomes of the parents. One part of the chromosome from each parent is then swapped, creating two children that each comprise of a selection of genetic material from both their parents.

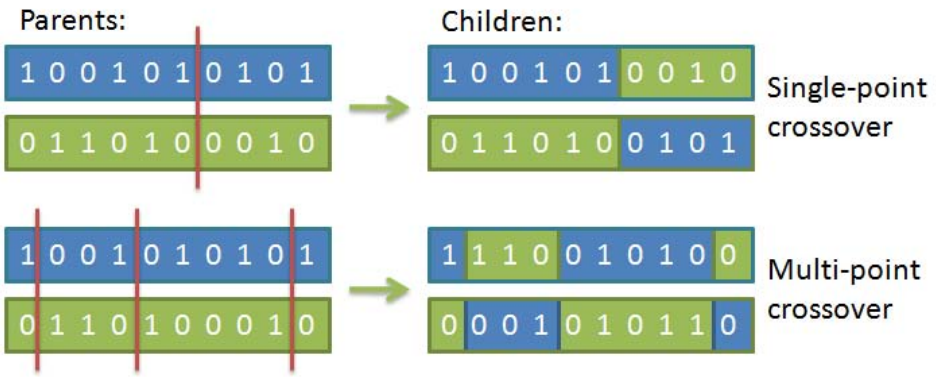

Figure 10 *Crossover (Recombination) methods for multi-point and single-point*

The method of multi-point crossover is similar to that of single-point, except that more than one split is made (figure 10). This has two advantages, the first being that the disruptive nature of multi-point crossover appears to encourage the exploration of the search space. The second is that parts of the chromosome representation that contribute the most to the performance of a particular individual may not necessarily be contained in adjacent strings [Booker 1987]. Other crossover methods include, uniform crossover, shuffle, and reduced surrogate operator [Mitchell 1996].

1.5 Mutation

Mutation is used to maintain the diversity of the entire population by changing individuals bit by bit with a small probability $pm \in [0,1]$, termed *mutation rate*. There is much debate whether high or low mutation rates should be used and whether these should be static or adaptive. A high mutation rate increases the level of exploration creating a more diverse population according to

[Michalewicz 2006], which is desirable for more complex combinatorial problems. However, there have been many proposed static mutation probabilities which are derived from experience or by trial-and-error. De Jong suggested pm = 0.001 in [De Jong 1975], with Schaffer *et al* extending this to a range of [0.001,0.005] [Schaffer 1989]. Bäck used Schaffer' 's results in [Back 1992] to propose that the mutation rate should be set according to population size and length of individuals, giving $pm = 1.75/(N*L^{1/2})$, where N is the population size and L denotes the length of individuals. Mühlenbein in [Muhlenbein 1992] recommended that $pm = 1 / L$ is an acceptable mutation rate and should be generally "optimal".

There is, however, evidence, both empirical [Fogarty 1989] for learning control rules, and theoretical [Back 1992] that the optimal rate of mutation is not only different for every problem but will vary with evolutionary time according to the state of the search and the nature of the landscape being searched. Recent work by Thierens [Thierens 2002] proposes two simple adaptive mutation rate control schemes called *constant gain* and *declining.* Thierens compares these to fixed mutation rates, and other known self-adaptive mutation rates showing that they perform favourably in terms of performance with no initial parameters to configure.

2. Genetic Algorithms and Real-Life Data Surfaces

GAs are shown to be compromised when directing search over significantly rugged surfaces [Goldberg 1992], such as those surfaces discussed in this work. As the amount of noise inherent in the surface increases, it is likely that the number of local optima increases and, unless there is sufficient diversity within the populations of the GA, this often causes the GA to converge on these local optima, rather than the global, optimal solution [Goldberg 1989]. Diversity is important in genetic algorithms as crossing over a homogeneous population does not yield new solutions [Fogel 2006]. The parameters of a GA can be improved for such problems, for example using a high mutation rate [Muhlenbein 1992], larger population sizes [Alander 1992] or by suitable selection techniques [Goldberg 1991]. *A priori* knowledge is typically required to set these parameters, although solutions such as adapting the parameters throughout the search using deterministic control schemes have been produced [Fogarty 1989, Baker 1985]. This section introduces the *random migration* operator based upon the *migration* operator that is used in multi-deme (multiple population) GAs [Mann 1999], and applies this to single-deme GAs supported by a decision support operator to yield a novel operator called *controlled migration*. It should be noted that controlled migration is equally applicable to the multi-deme case although this is not investigated here.

2.1 Random Migration

Multi-deme GAs make use of the migration operator to pass individuals between sub populations according to a pre determined migration rate and migration interval. During a search, sub populations will receive a new individual from another sub population that could be from anywhere in the global search space. The individual that is received is likely to have been evolved in a sub population that may be converging towards an alternative optimum, thus creating diversity in the receiving population. In a single-deme (single population) GA, a similar

scheme can be applied where random individuals are introduced into each generation from the global search space, thus introducing an alternative source of diversity. A typical GA will use either the incremental/steady state genetic algorithm (IGA) model [Whitley 1998] or the generational genetic algorithm (GGA) [De Jong 1992, Vavak 1996]. This section uses the GGA that batch replaces an entire population each generation, as opposed to the IGA which in typical applications only replaces one individual at a time. Figure 11 represents the GGA methodology that is applied in this section.

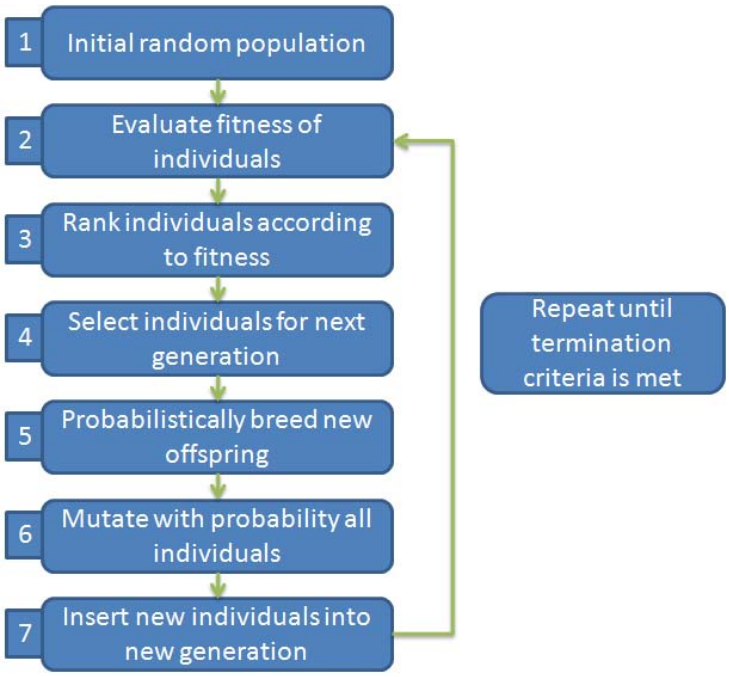

Figure 11 *GGA methodology applied in this section*

To insert random individuals into a generation, the following changes are necessary: In step 4 select fewer individuals that are required to create the next population; step 7 is then altered to insert the processed individuals from step 6 into a new generation, and to also introduce randomly generated individuals termed *migrants* to maintain the population size (figure 12). The term *migration rate* defines the number of migrants to insert into the new population, and hence the number of individuals to select in step 4 will be equal to the original population size less the migration rate.

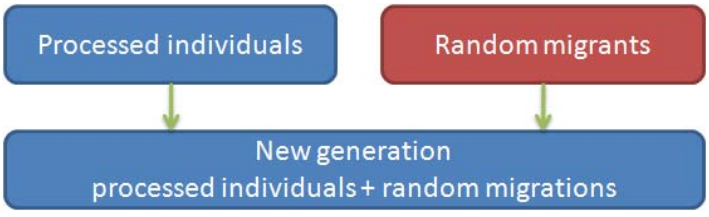

Figure 12 *New generation compiled of processed individuals and random migrants*

Using this methodology with the GGA parameters as declared in table 1, the range of surfaces *Peaks0* to *Peaks3* and *Bumps* are searched. The GGA is run 100 times per surface, producing mean results to negate the effects of the inherently stochastic heuristic.

Table 1 *GGA parameters used for search*

Parameter	**Value**
Max number of generations	Infinity (stop criteria used)
No. individuals in a population	20
Survival Selection	Stochastic Universal Sampling
Recombination method	Multi-point crossover
Crossover probability	0.7
Mutation rate	0.00875

Table 2 *Effects of increasing random migrants across the test surfaces with a population size of 20, (upper value: mean, lower value: worst case)*

Surface / **Random Migrants**	peaks0	peaks1	peaks2	peaks3	bump	Total
0	548 1012	1628 17934	10492 88650	255500 1290183	35552 408221	303720 1806000
2	553 1070	1088 5546	2447 25710	48113 363686	29028 160780	81229 597645
4	631 1145	1016 2401	1906 11743	17468 87780	38432 189995	59453 293064
6	682 1552	1306 4424	1937 7229	9310 53122	18475 107279	36982 173606
8	675 1744	1208 5685	2248 5303	7677 36729	12840 76894	24648 126355
10	804 2365	1580 4747	2417 10250	5259 28208	10455 56045	20515 101615
12	927 4065	1916 6688	2476 9364	8096 28825	9434 43765	22849 92707
14	1152 3842	2617 8566	6453 18447	5390 17123	7180 37563	22792 85541
16	1438 4822	3531 11502	11497 51503	12233 46523	9795 37062	38494 151412
18	4697 13662	13071 58641	31529 105001	48374 214102	12066 68060	109737 459466

Table 2 shows the effect on computations of inserting random migrants into a population of size 20 for a range of migration rates. A computation is counted as

each evaluation of an individual. It shows that introducing random migration for complex surfaces such as *peaks3* and the *bump* yields a considerable decrease in the number of computations compared to having no migrants. Figure 13 and figure 14 illustrate the effects of the different migration rates across these surfaces, clearly showing that increasing the migration rate reduces computations until a critical point where the search starts to degrade. A justification for this observation is that as the migration rate increases, then so does the diversity of the population with only a small number of highly ranked individuals surviving. As the migration rate nears the population size, then the search is comparable to a random search. Observing the results from the less complex surfaces it can also be seen that a critical point also exists, albeit to a lesser degree, where a random migration rate is present that increases the performance of the GA (Figure 15).

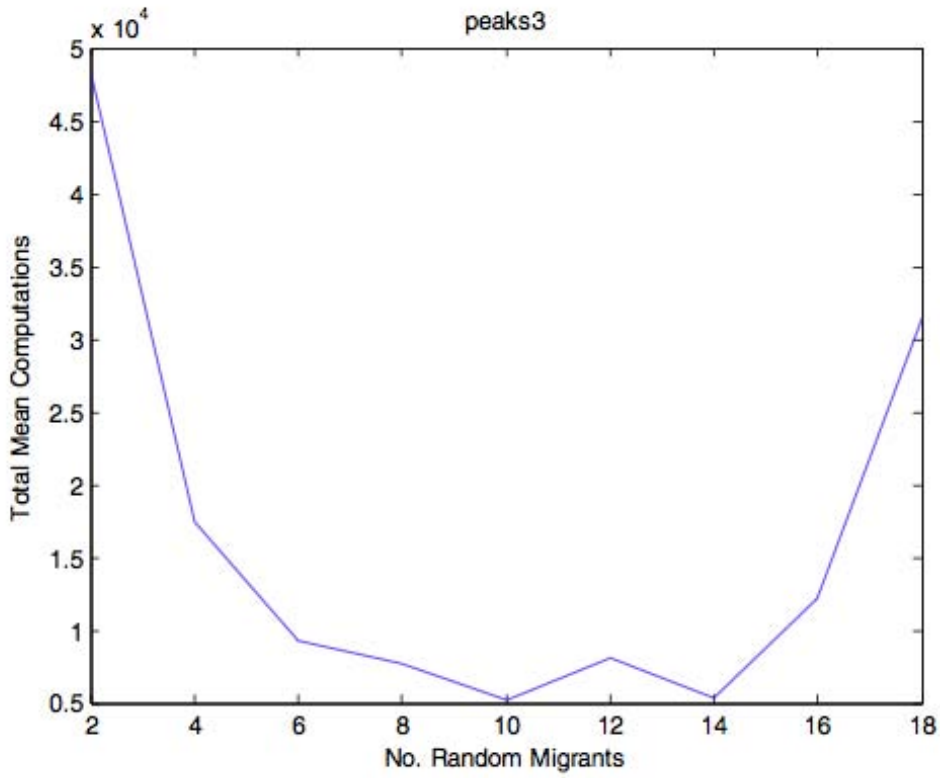

Figure 13 *Mean computations on peaks3 surface showing the effects of varying the number of random migrants for population of size 20*

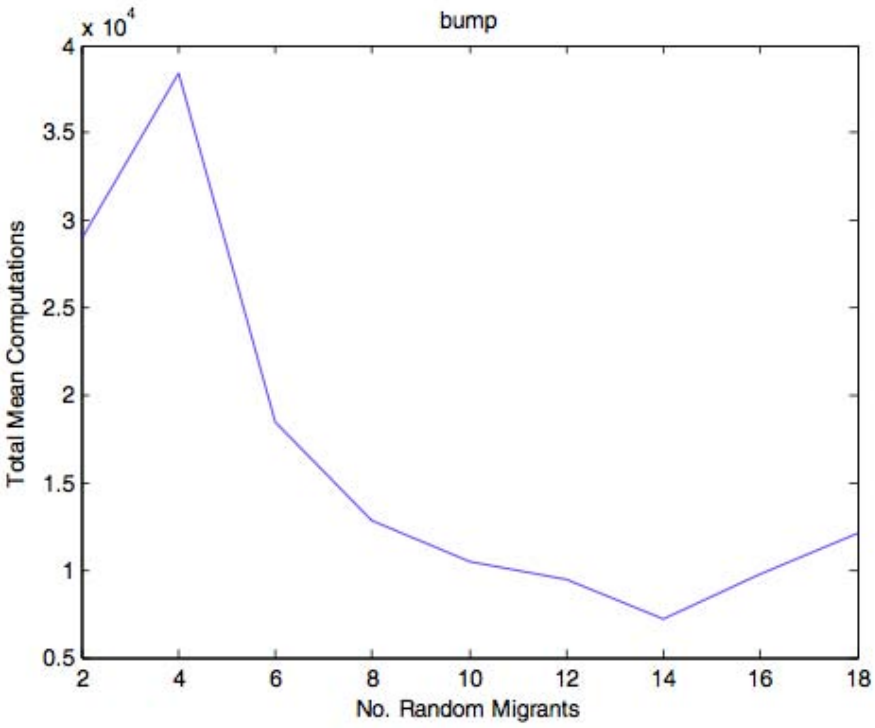

Figure 14 *Mean computations on bump surface showing the effects of the number of random migrants for population size of 20*

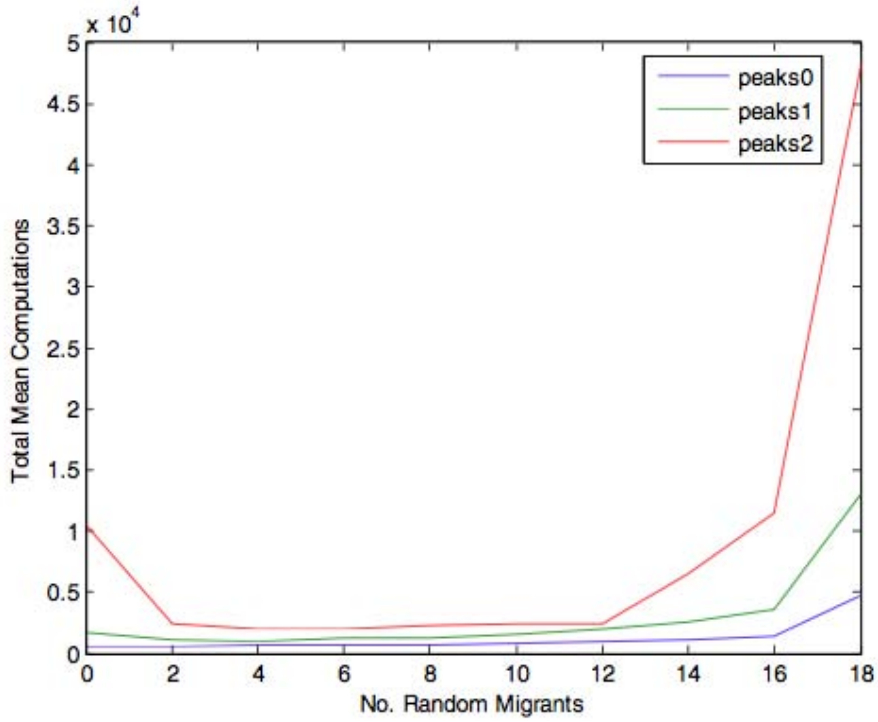

Figure 15 *Mean computations for peaks0, peaks1 and peaks2 showing the effects of the number of random migrants for population size of 20*

The results show how introducing random migration into single-deme GAs can

increase diversity, and hence lead to dramatic search improvements, particularly on rugged or complex surfaces. However, it is apparent that there is a critical migration rate that varies according to the complexity of the surface. A high random migration rate leads to excessive diversity analogous to high mutation rates, where previous work has shown a similar effect [Spears 1992]. It can be seen that low to mid random migration rates are a good trade-off between performance gains for complex surfaces, whilst minimising additional computation requirements for less complex surfaces. As with other genetic operators, the introduction of random migration has introduced another parameter that for improved effectiveness would require *a priori* knowledge of the surface to set a random migration rate. However, the next section will show that by applying the WSDS decision support to random migration, it is possible to minimise penalties for higher random migration rates on less complex surfaces.

Controlled Migration

The WSDS method introduced earlier in the chapter is used to provide decision support to random jumps in gradient decent methods. The same WSDS method can be applied to random migration to create a novel operator termed *controlled migration*.

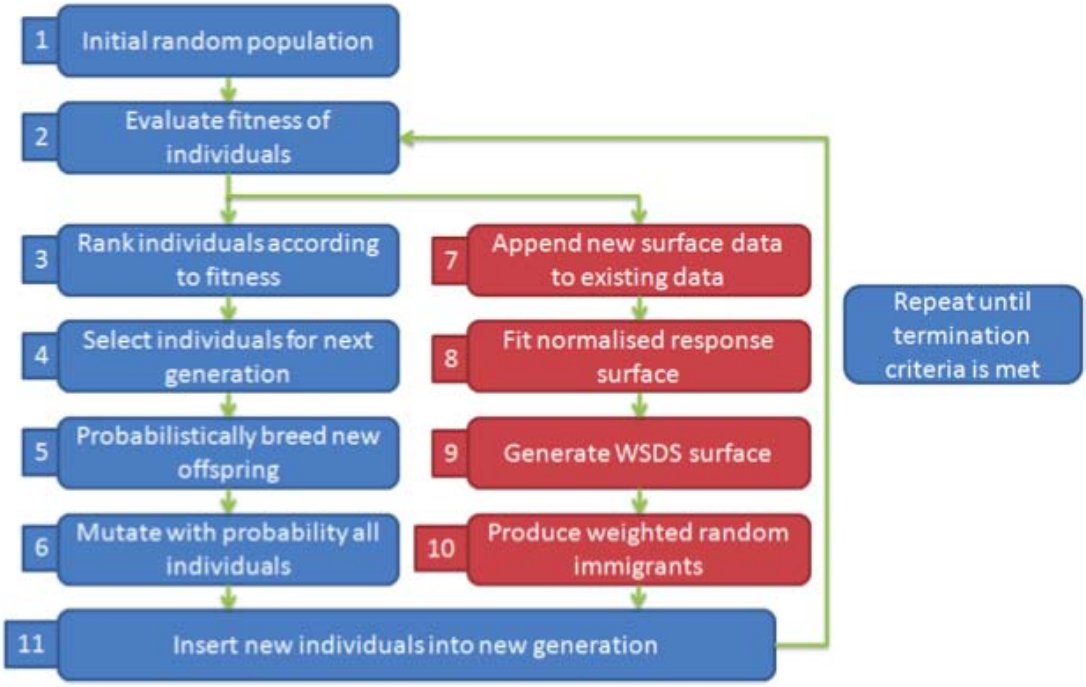

Figure 16 *GGA methodology with addition of WSDS random immigrants*

Figure 16 illustrates how the WSDS is integrated into the GGA methodology, with the additional steps coloured in red. During the first generation, the evaluation results from each individual in the population are used collectively to provide the data to fit the normalised response surface to. This response surface is then used to create the WSDS surface as defined in the accompanying paper. According to the controlled migration rate, a number of migrants are then

probabilistically selected from the WSDS surface and inserted into the new generation, along with the individuals processed by the standard GGA operators. This procedure is repeated for each generation, with the evaluation of each individual feeding into the data used to update the normalised response surface, thus the probability of selection of the next controlled migrants are based statistically on the results of the previous generations.

Table 3 *Effects of increasing controlled migrants across the test surfaces with a population size of 20*

Surface / **Controlled Migrants**	peaks0	peaks1	peaks2	peaks3	bump	Total
2	606 1445	900 4398	1307 4666	26578 229532	4806 22919	34197 262960
4	539 1136	1063 4755	1393 5464	12563 132622	5608 19422	21166 163399
6	624 1682	1091 3974	1877 9767	5641 29022	3015 10319	12248 54764
8	602 1503	1099 3485	1683 7448	5725 31526	1804 9133	10911 53095
10	647 1630	1425 4729	2244 9585	2879 15922	1628 6895	8823 38761
12	714 2268	1625 6721	3318 16723	3393 13108	1693 5566	10742 44386
14	941 3462	1644 6681	4580 16622	5275 26703	1254 4450	13694 57918
16	989 4601	2677 9222	10897 32281	5411 22762	978 4983	20952 73849
18	2840 11727	7253 29322	35623 162742	17927 101061	1244 4799	64887 309651

Using the GGA parameters presented in table 1, the experiment from the previous section is repeated replacing the random migrants with controlled migrants using the prescribed method for a range of controlled migration rates. Based on the results of the WSDS methodology applied to gradient descent methods, a 2nd order support surface is chosen.

The search is conducted running the GGA on each surface 100 times to yield the mean and maximum number of computations as shown in table 3. From the results it is immediately evident that using the controlled migration gives a ~35% improvement in performance at migration rates of interest compared to using random migration. Figure 17 and figure 18 illustrate this improvement using the total mean and max computations respectively across all the surfaces. Comparing the totals across all surfaces is justified, as although the more complex surfaces contribute most to the improvements observed, there are no

significant declines in performance for less complex surfaces. Moreover, controlled migration appears to minimise penalties for higher migration rates on the basic surfaces (figure 19 and figure 20). This further endorses the generality of controlled migration as a viable operator to speed-up GA searches, as whilst it appears to cause no significant detrimental affect, it can provide a major performance boost on such surfaces as the bump (figure 4.27).

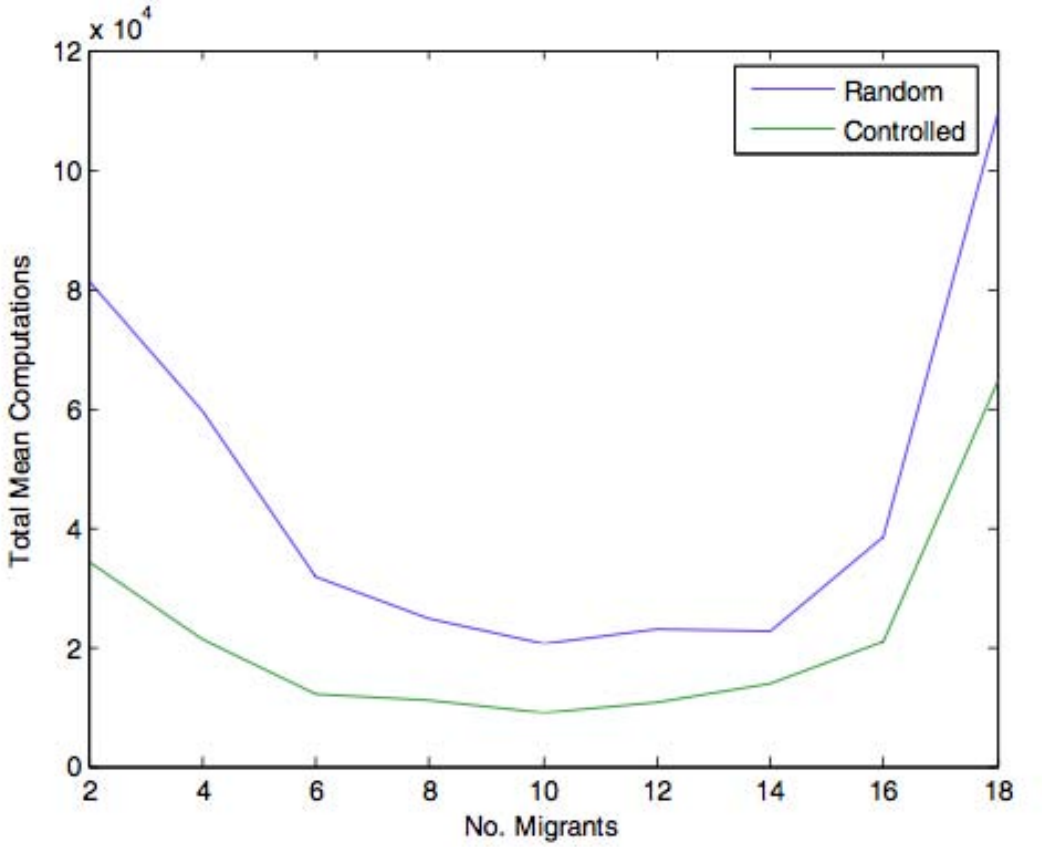

Figure 17 *Comparison of random migration vs. controlled migration for total mean computations across all surfaces*

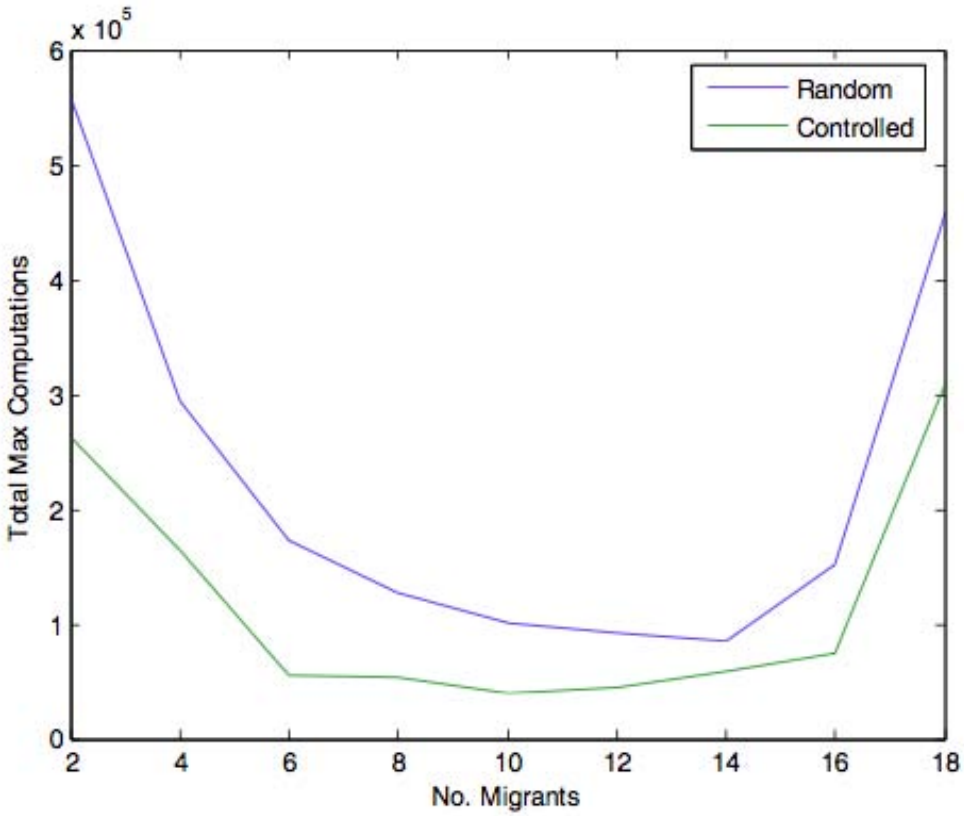

Figure 18 *Comparison of random migration vs. controlled migration for total maximum computations across all surfaces*

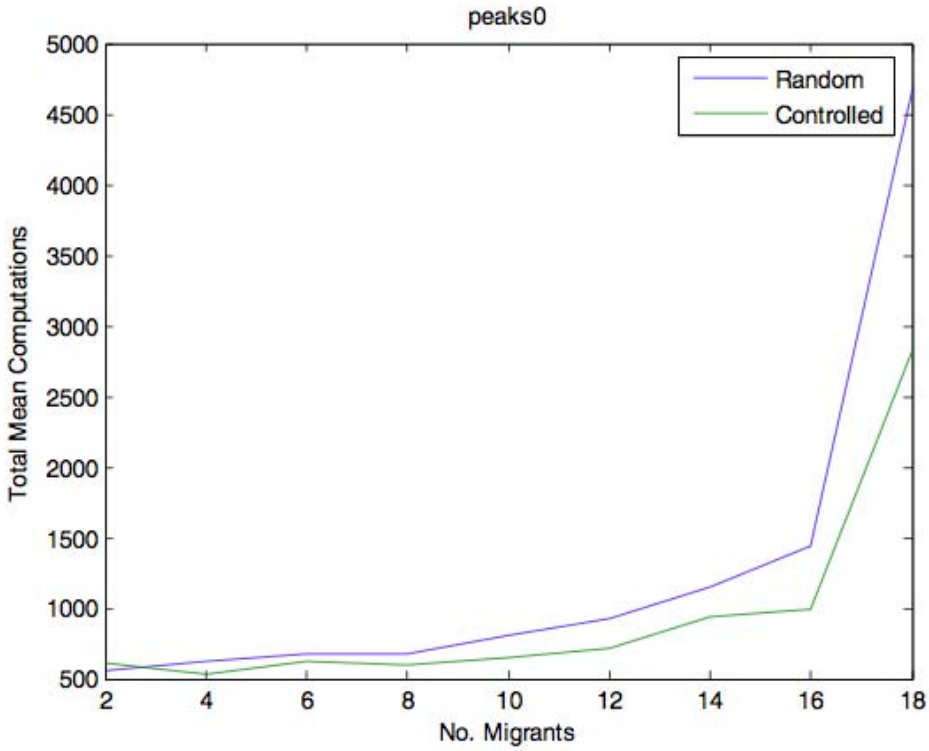

Figure 19 *Comparison of random migration vs. controlled migration for mean computation on peaks0*

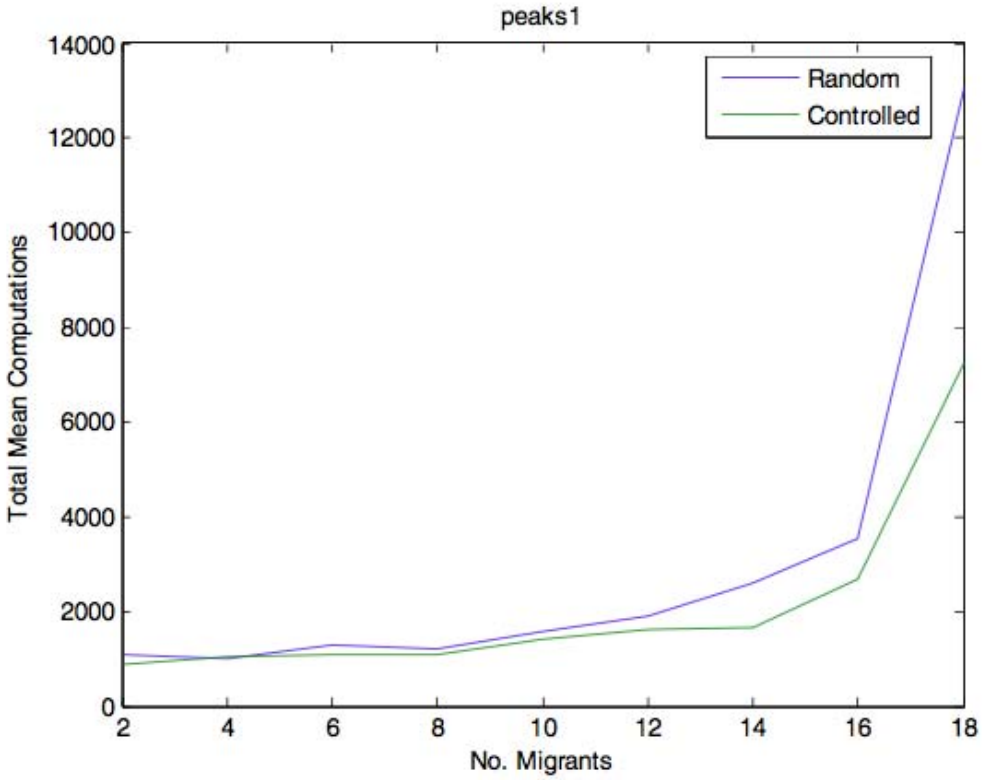

Figure 20 *Comparison of random migration vs. controlled migration for mean computation on peaks1*

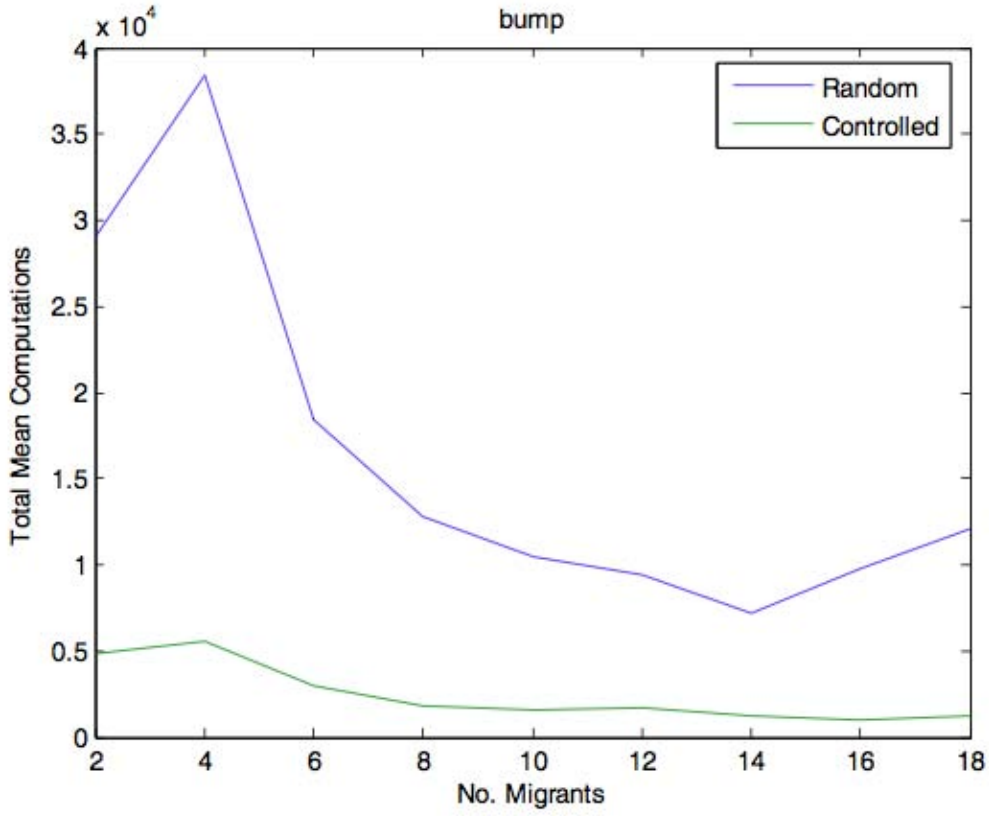

Figure 21 *Comparison of random migration vs. controlled migration for mean computations on bump*

Conclusions

This report introduces a novel decision support methodology based upon response surfaces. The method had been previously applied to add decision support to the previously random jumps of gradient descent methods commonly used in combinatorial experimentation [Jaskiewiscz 1998]. The response surfaces are generated through exploitation of evaluated data that is performed during the search that would, without the use of metaheuristics, be discarded. These response surfaces are transformed into normalised contours of the search area, providing weighted stochastical decision supported jumps. The analysis had previously compared support surfaces of varying orders against five separate surfaces varying in complexity, using three different search heuristics.

The decision support methodology is now applied to GAs, first investigating a mechanism for the introduction of a decision supported operator. Migration in single-deme GAs of random individuals was investigated as an alternative to mutation as a means of maintaining diversity in each generation. Random migration is then demonstrated to provide substantial improvements in the efficiency of a GA when faced with more complex or rugged surfaces that contain many local optima. Moreover, this migration operator provides the mechanism for which to apply decision support, and is introduced as controlled migration. Through a comparison with random migration, controlled migration is shown to provide an improvement in required computations by up to a factor of two. With both migration operators, it is apparent that there is a critical migration rate that varies according to the complexity of the surface. A high migration rate leads to excessive diversity analogous to high mutation rates. A low migration rate, whilst providing minimum risk for computation penalties for simple surfaces, does not exploit the benefits attainable when applied to more complex surfaces. However, using controlled migration over random migration is shown to allow higher migration rates, whilst minimising the detrimental effects on simple surfaces. This can be explained as whilst exploring simple surfaces, the decision support surface is more likely to provide a reliable estimate as to where the minimum or maximum lies. Using high controlled migration rates, it is more probable that good candidate individuals are chosen.

Next Steps

Now that a methodology for dealing with noisy sensor data has been developed, Prof Stewart will investigate the development of automatic model generation, and its integration with the newly developed *Controlled Migration* methodology, to complete an integrated environment to process multi source, noisy, sensor data.

Bibliography

Alander T., On optimal population size of genetic algorithms, in Proc. IEEE Int. Conf. Computer Systems and Software Engineering, pp. 65-70, 1992.

Bäck T., Self-adaptation in genetic algorithms, In Proc. of the 1st European Conference on Artificial Life, pp. 263-271, 1992.

Baker, J. E., Adaptive Selection Methods for Genetic Algorithms. In Proceedings of the 1st international Conference on Genetic Algorithms J. J. Grefenstette, Ed. L. Erlbaum Associates, Hillsdale, NJ, pp. 101-111, 1985.

Baker J. E., Reducing bias and inefficiency in the selection algorithm. Proceedings of the Second International Conference on Genetic Algorithms and Their Applications, Cambridge, MA, Erilbaum, pp. 14-21, 1987.

Booker L., Improving search in genetic algorithms, In Genetic Algorithms and Simulated Annealing, Davis L., (Ed.), pp. 61-73, Morgan Kaufmann Publishers, 1987.

Bramlette M. F., Initialization, Mutation and Selection Methods in Genetic Algorithms for Function Optimization, Proc ICGA, Vol. 4, pp. 100-107, 1991.

Burke E. K., and Smith A. J., Hybrid evolutionary techniques for the maintenance scheduling problem, Power Systems, IEEE Transactions on , Vol. 15, No. 1, pp. 122-128, Feb 2000.

Caruana R. and Schaffer J., Representation and Hidden Bias: Gray vs. Binary Coding for Genetic Algorithms, In Laird, J. (ed.) Proceedings of the Fifth International Conference on Machine Learning, Morgan Kaufmann Publishers. San Mateo, CA, pp. 153-161, 1988.

Chipperfield J., Fleming P. J., The MATLAB Genetic Algorithm Toolbox, IEE Colloqium on Applied Control Techniques Using MATLAB, Digest No. 1995/014, 26 Jan 1995.

De Jong K. A., An analysis of the behaviour of a class of genetic adaptive systems. PhD Thesis, Department of Computer and Communication Science, University of Michigan, Ann Abor, 1975.

De Jong K. A., Are genetic algorithms function optimizers? Parallel Problem Solving From Nature 2, Elsevier Science Publisher, 1992.

Fogarty T. C., Varying the Probability of Mutation in the Genetic Algorithm, Proceedings of the 3rd International Conference on Genetic Algorithms, pp.104-109, June 01, 1989.

Fogel D. B., Evolutionary Computation: Toward a New Philosophy of Machine Intelligence, New York: IEEE Press, 2006.

Fonseca C. M., and Fleming P. J., Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization, in Proceedings of the Fifth International Conference on Genetic Algorithms, Forrest S., Ed. San Mateo, CA, Morgan Kauffman, pp. 416 – –423, 1993.

Fonseca C. M., and Fleming P. J., Multiobjective optimal controller design with genetic algorithms, International Conference on Control, Vol.1, pp. 745-749, 21-24 March 1994.

Goldberg D. E., Genetic algorithms in search, optimization and machine learning, Addison-Wesley, Reading, MA, 1989.

Goldberg D. E., and Deb K., A Comparative Analysis of Selection Schemes used in Genetic Algorithms. Foundations of Genetic Algorithms. San Mateo, CA, Morgan Kaufmann 1, pp. 69–93. (Also TCGA Report 90007), 1991.

Goldberg D. E., Deb K., and Clark J. H., Genetic algorithms, noise, and the sizing of populations, Complex Systems, Vol. 6, pp. 333-362, 1992.

Holland J. H., Adaptation in natural and artificial systems, The University of Michigan Press, Ann Arbor, 1975.

Jaszkiewicz A., Genetic local search for multiple objective combinatorial optimisation. Technical Report RA-014/98, Institute of Computing Science, Poznan University of Technology, December 1998.

Keane A. J., Experiences with optimizers in structural design, Proceedings of the Conference on Adaptive Computing in Engineering Design and Control ' '94, ed. I.C. Parmee, Plymouth, pp. 14-27, 1994.

Keane A. J., Genetic algorithm optimization of multi-peak problems: studies in convergence and robustness, Artificial Intelligence in Engineering, Vol. 9, No. 2, pp. 75-83. 1995.

Koza J. R., Keane M. A., Yu J., Bennett III F. H., Mydluwec W., Automatic Creation of Human-Competitive Programs and Controllers by Means of Genetic Programming. Genetic Programming and Evolvable Machines, Vol. 1, No. 1, 121- 164, April 2000.

Man K. F., Tang K. S., Kwong S., and Halang W. A., Genetic Algorithms: Concepts and Designs, Springer-Verlag: London, pp. 46-50, 1999.

Michalewicz Z., and Fogel D. B., How to solve it, algorithms for engineering systems, Cambridge University Press, Cambridge UK, ISBN-13-978-0-521-85564-8, 2006.

Mitchell M., An Introduction to Genetic Algorithms, MIT Press, 1996.

Mühlenbein H., How Genetic Algorithms really work. Mutation and hill-climbing. In: R. Männer and R. Manderick, Editors, Parallel Problem Solving from Nature PPSN II, North-Holland, Amsterdam, pp. 15 – –25, 1992.

Schmitendorgf W. E., Shaw O., Benson R., and Forrest S., Using Genetic Algorithms for Controller Design: Simultaneous Stabilization and Eigenvalue Placement in a Region, Technical Report No. CS92-9, Dept. Computer Science, College of Engineering, University of New Mexico, 1992.

Spears W. M., Crossover or Mutation? In Whitley (ed.) Foundations of Genetic Algorithms-2. pp. 221-237. Morgan Kaufmann Publishers. San Mateo, CA. 1992.

Stewart P., Gladwin D., Stewart J., Chen R. and Winward E., ""Improved decision support for engine-in-the-loop experimental design optimization" ", Institute of Mechanical Engineers Part D - Automobile Engineering. Vol. 224. (In Press).

Thierens D., Adaptive mutation rate control schemes in genetic algorithms, IEEE International Conference on E-Commerce Technology, Evolutionary Computation, Vol. 1, pp. 980-985, 2002.

Vavak F., and Fogarty T. C., A Comparative Study of Steady State and Generational Genetic Algorithms, Selected Papers from AISB Workshop on Evolutionary Computing, pp.297-304, April 01-02, 1996.

Whitley D., and Kauth J., GENITOR: A different Genetic Algorithm, in hoc. of the Rocky Mountain Conf. on Artificial Intelligence, Denver. 1998.

Wright A. H., Genetic Algorithms for Real Parameter Optimization, In Foundations of Genetic Algorithms, J. E. Rawlins (Ed.), Morgan Kaufmann, pp. 205-218, 1991.

Sensor Fusion, Prognostics, Diagnostics and Failure Mode Control for Complex Systems – 12 Month Report

Professor Paul Stewart
Head of School of Engineering
University of Lincoln UK

Overview

In the first (6-month) report, a novel search methodology is presented to support the operation of problem solving heuristics when dealing with real-life noisy data measurements. In this (12-month) report, a novel methodology is presented to accelerate the process of control system design with hardware in the loop. A representative aircraft system is considered on a real experimental rig which produces noisy data streams. The combination of methods is shown to give significant acceleration in identifying acceptable controllers for the flight control system.

Introduction

For hardware (aircraft)-in-the-loop searches it is desirable, essential even, that the number of computations (simulations or hardware-in-the-loop trials) is minimised. Methods of speed-up are therefore required to either enable parallelisation of the search process or to minimise the number of computations required.

This report details how the computational framework is designed to be distributed across multiple processors, allowing computations to be executed in parallel using a choice of two strategies for speed-up. It then discusses how controllers can be automatically created with hardware-in- the-loop. The performance of a particular control solution implemented in hardware is fundamentally based upon the accuracy of the model from which it has been derived. It is often the case that controllers that have been produced through simulation require further tuning in hardware to achieve acceptable results. The substitution of hardware-in-the-loop for the software model has been shown to introduce a degree of robustness that is not easily achieved under simulation **[1].** This is due to the hardware setup possessing disturbances such as noise on sensors and the inclusion of unknown plant dynamics that unless pre-empted would not be included in the software model **[2]**. However, hardware experiments are likely to take much longer to evaluate than simulations, making iterative searches involving hundreds or thousands of trials often unfeasible. A hybrid approach is presented in this report using hardware-in-the- loop and software simulations, with the aim of carrying out the least number of experiments on the hardware. This is shown via a designed experiment to be successful at exploiting the advantages of using hardware-in-the-loop, whilst minimising the number of actual hardware experiments.

1. Search System Experimental Configuration

The search system server components, Apache and MySQL, are installed on a master machine. The MySQL database maintains a list of slave machine IP

addresses that are allowed to communicate with the master, along with the type of slave (simulation or hardware) they are operating as. Once a job is instigated on the master, using MATLAB scripts the slaves initiate requests for work; this method is termed pull technology as opposed to push technology, where the master would send work to the slaves **[3].** The slaves repeat requests for work until the master responds with data which can contain Simulink files, data arrays, and MATLAB scripts. Transfer of data is either through the MATLAB interface, or via the master transferring files into shared folders on the slaves. Once a slave completes its job, the results are stored directly into the MySQL database and the slave initiates another request for work.

Figure 1 – Photo showing experimental setup, master machine and screen on the left and four simulation machines on the right connected via KVM to single screen

The simulation based experiments in this report are all carried out using five 2.0GHz Intel Core 2 Duo E4400 machines with 2GB RAM (figure 2) connected over 1GB/s Ethernet. Although only five machines are presented here (one master, four simulation slaves), all the ideas and strategies introduced in this chapter are scalable. In fact, as the slave machines use pull technology, additional slaves can be added by simply allowing the master to communicate with them. It is possible to search for solutions in real-time with hardware-in-the-loop rather than software simulation using models. This is achieved using National Instruments LabVIEW **[4]** hardware and dedicated software installed on an additional machine. LabVIEW is configured to provide an interface in real-time between Simulink and the hardware to be controlled. In the setup described, this additional 6th machine is termed a *hardware slave*, and is connected up to the hardware under test. Again, it is possible to have more hardware slaves if multiple hardware apparatus is available. As will be discussed later, hardware slaves are treated as special cases; they request and receive work in the same manner, but work is only given to them if a hardware test is required. Figure 3 shows an overview of the complete setup; it should be noted that as the communication is over standard Ethernet, the components of the system do not need to be physically located together. In fact, the configuration used for

hardware-in-the-loop experiments in this chapter are carried out with the LabVIEW system located remotely. Due to the run time of most hardware experiments, the communication overheads are negligible for this size of system.

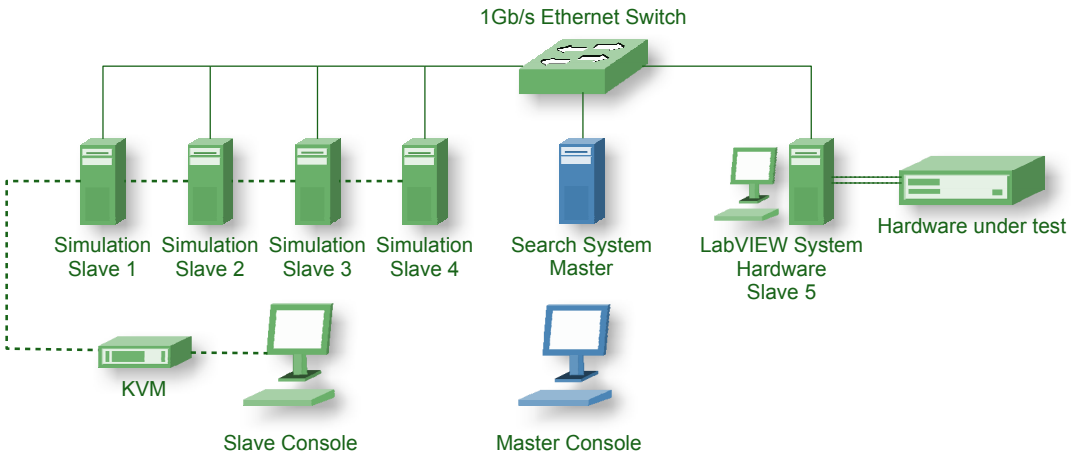

Figure 2 – Overview of search system setup for parallel and distributed computations

2. Hardware Based Search

It is common for simulated models of hardware to be used to design controllers. However, it has been demonstrated by Stewart **[1],** that even in cases where the model is of satisfactory accuracy, implementing the developed controller in hardware gives differing results. It is often the case that further manual tuning of the controller in real-time is required to match or better the results obtained under simulation. Automatic tuning of a controller with hardware-in-the-loop has been shown to create successful controllers.

Stewart **[1]** has shown through the method of tuning a fuzzy logic controller by a GA with hardware-in-the-loop, that this can produce controllers with a higher level of robustness than can be achieved under simulation **[5].** The penalty for such a method is noticeably that of the time required to execute each iteration of the search algorithm. Designed experiments with hardware-in-the-loop are more likely to take longer to process than simulations in software. It is assumed in most scenarios that it is unlikely that multiple hardware slaves are available, either due to cost or complexity and hence parallel trials are not feasible. This section, therefore, investigates a search methodology to search for controllers with hardware-in-the-loop that minimises the number of experiments. Control solutions are expected to perform as anticipated when implemented in hardware, and exhibit levels of robustness attributed to typical hardware-in-the-loop tuning methods.

2.1 Pre-selection for Hardware-In-The-Loop

In the case of the search system introduced in this report, where controller topologies are automatically created, the number of experimental runs is likely to be large. Using hardware-in-the-loop for all fitness evaluations is expected to be unfeasible. It is therefore desirable to create a hybrid search strategy whereby hardware experiments and simulations are used in combination. This can be achieved by primarily using a model in simulation to drive the search, but using the hardware to validate and correct the evaluations. The simulations are effectively filtering out candidate solutions that are known to have poor performance, and hence avoid any unnecessary hardware

experiments. This methodology will be referred to as *pre-selection*.
Pre-selection is implemented in the search system as illustrated in figure 3.9. Each individual is evaluated in simulation and its fitness calculated according to the objective functions. If the fitness of an individual is ranked higher (performs better) than a set threshold, these individuals are then re-evaluated with hardware-in-the-loop. The experimental results replace the simulation results and the fitness of the individual is recalculated. Individuals that have fitness values ranking lower than the threshold are not evaluated with hardware, and their fitness value remains.

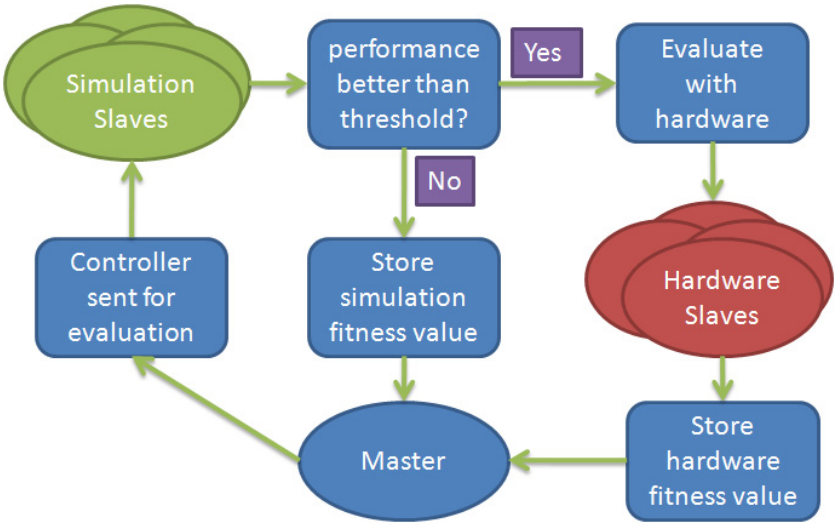

Figure 3 – Pre-selection method to minimise hardware-in-the-loop experiments

Individuals requiring hardware-in-the-loop trials are put into a queue in the search system in order of fitness ranking. This queue is then processed by one or more hardware slaves, with the expectation that by processing the better performing individuals first, it is more likely that a solution meeting the stop criteria will be found earlier. As the hardware experiments are executed on dedicated hardware slaves, simulations of individuals can continue in parallel until all individuals in the current population have been evaluated. Following the fitness evaluation of all the individuals in the population, ranking is then performed on this hybrid of data in the conventional manner.

2.2 Hardware Experimental Setup

To demonstrate the search system with hardware-in-the-loop using pre-selection, this section will describe an experimental setup for an Electric Thrust Reversal Actuation System (ETRAS). The More Electric Aircraft (MEA) is a term given to replacing conventional systems utilising hydraulic and pneumatic systems within the engine and airframe, with electric systems. An example of research and development in this area is the Reverse Thrust System. This provides a means of decelerating aircraft during landing, thus reducing the landing distance required. Traditional Reverse Thrust Systems are hydraulically operated, research has been carried out to find an all-electric replacement. A system known as the Electric Thrust Reversal Actuation System (ETRAS) is now commercially available and is fitted to both power plants offered for the new Airbus A380 super-jumbo.
The ETRAS has the advantage of being lighter, cost effective, more reliable and

easier to maintain than its hydraulic counterpart. The ETRAS is implemented through actuators deploying a cowl that acts to deflect the flow of air through side panels that open up around the engine. This causes the air to flow out of the side openings in the opposite direction creating Reverse Thrust.
A proof-of-concept demonstration rig has been previously developed to investigate the control system required to deploy the cowls. The rig comprises three pairs of DC motors, with each pair consisting of a drive motor connected to a load motor via a threaded shaft. The drive motor is used to rotate the shaft, providing linear displacement of the cowl, while the load motor applies time varying loads to the shaft. The control system is designed to deploy and retract the cowls within a specified time frame, and to ensure that the position of the three cowls are always synchronised. The DC motors are controlled using Pulse Width Modulation (PWM), with each motor having its own controller (Figure 4). The system is non-linear due to the dump circuit used to dissipate the regenerated power, and uses a power supply of limited current restricting the maximum torque of the DC motors.

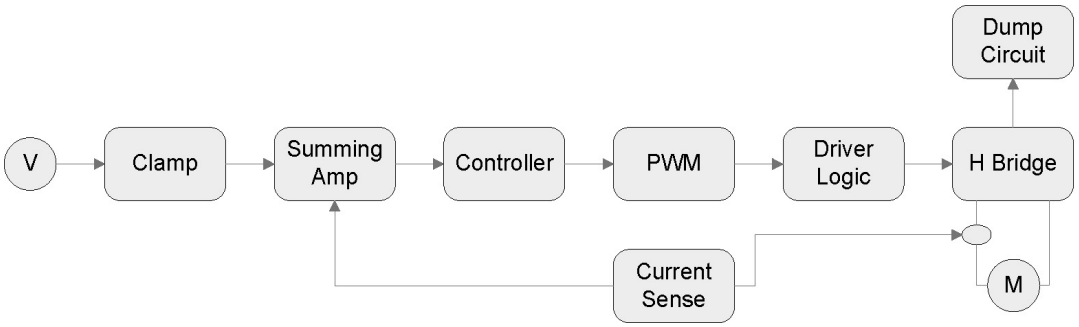

Figure 4 – Block diagram of DC motor electronics

The ETRAS test rig is modelled through designed experiments as a non-linear system to create the Simulink file as shown in figure 5 representing an individual drive motor. The Simulink file is a model of the ETRAS rig controlled via the LabVIEW system. However, the model has been simplified to ensure it is not a precise representation of the hardware to assess whether the search system can converge on a solution.

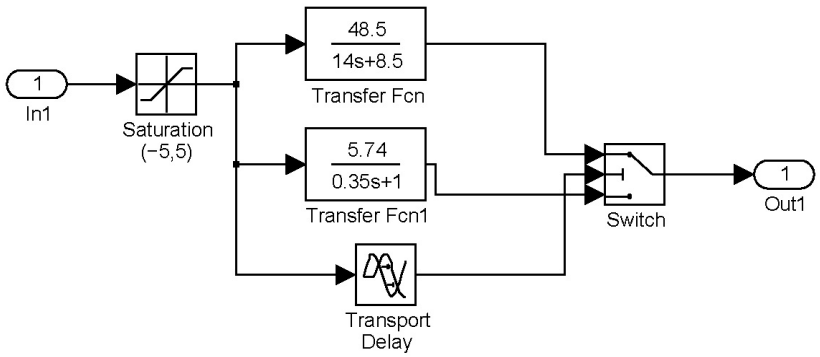

Figure 5 – Simulink model of experimental rig

The search system is used to automate the design of a speed controller for an individual drive motor. The input demand is a pulse from 0 rad/s to 180 rad/s with a time period of 10.5s and 50% pulse width. The search system is configured to track

the demand with two objectives; to minimise overshoot to less than 0.5%, and to minimise the integral of time multiplied by the absolute value of error (ITAE) to less than 15%. The objective for the ITAE is lenient in this case due to the restriction in rise time caused by a limit on the power supply current.

2.3 Pre-selection Analysis and Results

To test the pre-selection methodology, the following experiment is conducted. The search system is configured with the parameters in table 1, these values are different to those used in previous searches thus far. The reason for this is to reduce the worst case running time of the search. The outer loop generations are reduced, along with the population sizes for both loops. The number of generations for the inner loop are increased and this decision is based on experience that if you decrease the population size, then for parameter tuning, the number of generations should be increased to enhance the likely hood of finding suitable solutions. The search system is then used with simulation evaluations only to find control solutions that meet the objective functions. From the good candidate solutions, the controller that provides the desired performance is manually chosen from observing the model response (figure 6). This controller is then implemented on the ETRAS rig using the LabVIEW setup; the response of the system is shown in figure 7. It can be seen that the performance of the controller in hardware does not match that achieved under simulation. This is expected since the simulation is performed on a simplified model, with differences highlighted by the maximum acceleration achieved at low speeds. The next step in the experiment is to run the search with hardware-in-the-loop using the pre-selection methodology, to establish whether a controller of higher quality can be found.

Parameter	**Value**
Search type	Continuous template search
Block set	Block set A
Outer loop:	
Max number of generations	20
No. individuals in a population	15
Inner loop:	
Max number of generations	30
No. individuals in a population	15
No. of random immigrants	0
Applicable to both loops:	
Survival Selection	Stochastic Universal Sampling
Recombination method	Multi-point crossover
Crossover probability	0.7
Mutation rate	0.00875

Table 1 – Search system parameters for ETRAS rig

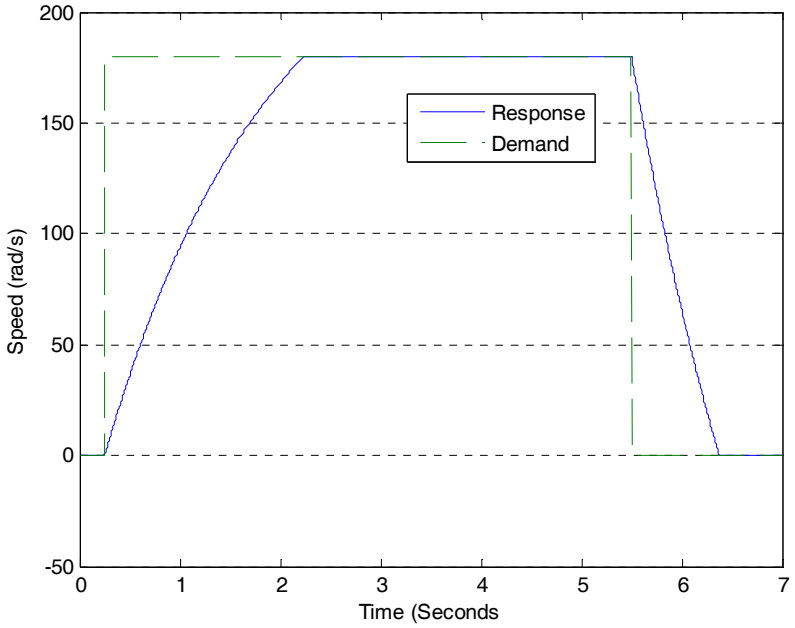

Figure 6 – Simulation response of best performing controller

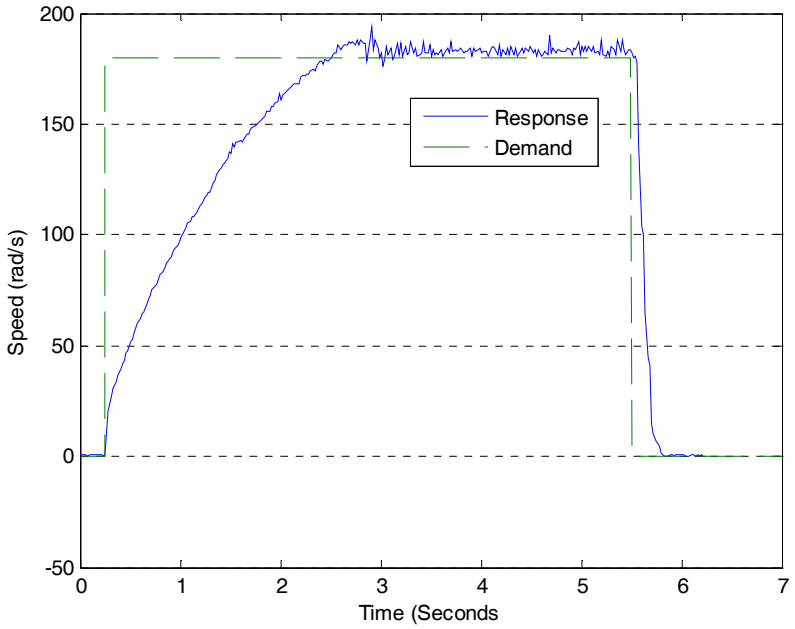

Figure 7 – Response of best performing controller implemented on hardware

The search is a multi objective search with two objectives, and therefore two threshold values are required. Through trial-and-error, the threshold values are established, observing whether the search is converging. It is reasonable to assume that the hardware should begin to be used within a few generations of the start. The rationale for this is that populations in early generations are likely to be very diverse; exploiting this opportunity means that the hardware evaluates additional global points. To establish suitable threshold values, a number of shorter searches are performed to observe the usages of the hardware. Through trial and error, a threshold value for overshoot is set at 4% and an ITAE of threshold value of 18%. The hardware is observed to be used between generation 5 and 7 on average using these values. Having set the threshold values, the search is now initiated with hardware-in-the-loop

using the parameters in table 1.

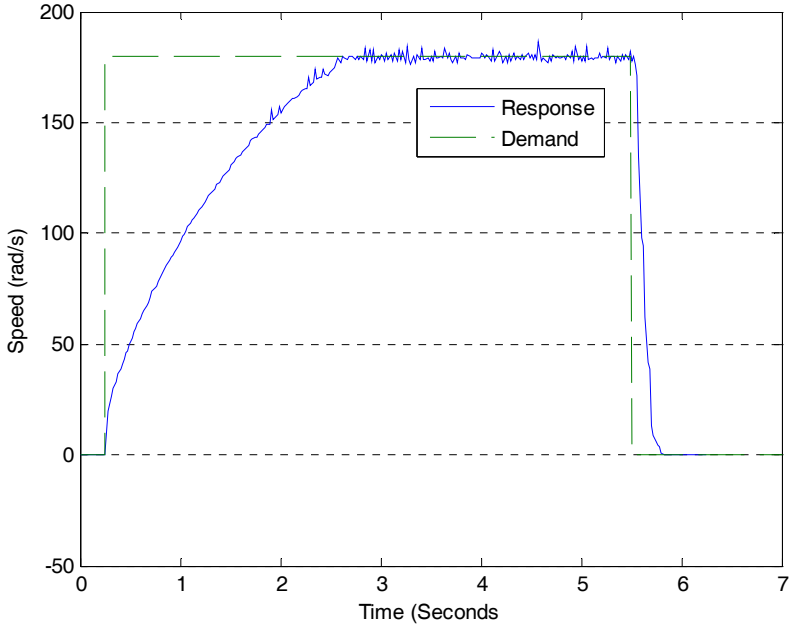

Figure 8 – Response of best performing controller found with hardware-in-the-loop

The response from the best controller (having the lowest objective functions), is implemented in hardware and shown in figure 8. Comparing this to figure 7, it is immediately evident that the new controller tracks the demand more accurately. The search was a continuous exploration, and as such the system carried out the maximum 180,000 simulations, with 11,301 fitness values being replaced by hardware experiments. This represents approximately 6.3% of all individuals performing better than the threshold values. It should be noted that the hardware experiments were evenly spread across the generations in the second half of the search. This is most likely due to the conservation of diversity across the outer loop generations, meaning that bad topologies that cannot be tuned successfully exist in each generation.
To conclude, the pre-selection methodology has been shown to reduce the number of hardware-in-the-loop experiments. In the case represented here, it would have taken approximately 34 days to carry out the search solely with hardware experiments, the actual runtime with pre-selection is just longer than 2 hours.

Conclusions

This report is concerned with the speed-up of search for hardware- in-the-loop based searches. The details for carrying out parallel and distributed computations with the search system are conveyed and used to perform large searches. Two methods of parallelisation are available due to the two loop feature of the search. In the first method, the master processes each iteration of the outer loop and subsequent inner loops serially. The slaves are used to evaluate the fitness values of the inner loop individuals in parallel. The result of this arrangement is that the processing is a form that is analogous to standard parallel genetic algorithms, and as such existing parallel methodologies can be used. The second method involves the master processing the outer loop only; the inner loop for each outer individual is processed in its entirety by a slave. Hence, outer loop individuals are processed in parallel. The advantage of this second method is that communication overheads are reduced by a factor of $G_I . X_I$.

This is because communication between master and slaves only occurs at each iteration of the outer loop, rather than at each iteration of the inner loop, as in the first method.
A methodology for minimising the number of trials when using hardware-in- the-loop to direct the search is presented. An experiment is conducted demonstrating that control solutions developed using models under simulation do not always perform as expected in hardware. The demonstration shows a controller developed from a simplified model for an Electric Thrust Reversal Actuation System (ETRAS), that when implemented in hardware does match the performance under simulation. This is because the controller is fundamentally based upon the accuracy of the model from which it has been derived. It is sometimes the case that very accurate models are not available and therefore it is desirable to use hardware-in-the-loop to direct the search. The main problem with hardware-in-the-loop is that performing hundreds or thousands of trials is often unfeasible due to time constraints.
To minimise the number of trials needed, *pre-selection* is introduced that uses a hybrid of simulation and hardware-in-the-loop trials to direct the search. The fitness of each individual is calculated using simulations in the first instance; if an individual performs better than a set threshold, then its fitness value is re-evaluated with hardware-in-the- loop. The search algorithm then uses the hybrid of fitness data to rank the results. The ETRAS controller is developed using pre-selection, showing that a controller can be found that matches the performance requirements. The threshold values were established through trial and error, through varying the values and observing whether the search was converging. Ultimately, out of 180,000 fitness calculations carried out under simulation, only 11,301 (6.3%) were re-evaluated with hardware-in-the-loop. This is a significant reduction in experimental trials making hardware-in-the-loop searches feasible.

Bibliography

[1] Stewart P., Stone D. A., and Fleming P. J., Design of robust fuzzy-logic control systems by multi-objective evolutionary methods with hardware in the loop, IFAC Journal of Engineering Applications of Artificial Intelligence, Vol. 70, No. 3, pp. 275-284, May 2004.

[2] Stewart P., Stone D. A., Fleming P. J., On-line multiobjective automatic control system generation by evolutionary algorithms, IEEE International Symposium on Industrial Electronics, Vol. 2, pp. 1459-1464, 4-7 May 2004.

[3] Martin-Flatin J. P., Push vs. Pull in Web-Based Network Management. In M. Sloman, S. Mazumdar, and E. Lupu (Eds.), Proc.6th IFIP/IEEE International Symposium on Integrated Network Management (IM' '99), Boston, MA, USA, May 1999, pp. 3 – –18. IEEE, Piscataway, NJ, USA, 1999.

[4] National Instruments, LabVIEW measurement and automation software, http://www.ni.com/labview/, accessed 28 March, 2009.

[5] Chandrasekharan P., C., Robust Control of Linear Dynamical Systems, Academic Press, 1996